Re-Factored Operational Support Systems for the Next Generation Platform-as-a-Service (NGPaaS)

Paul Veitch[1], Adam Broadbent[1], Steven Van Rossem[2], Bessem Sayadi[3], Lionel Natarianni[3], Bilal Al Jammal[3], Laurent Roullet[3], Angelos Mimidis[4], Eder Ollora[4], Jose Soler[4], Sebastien Pinniterre[5], Michele Paolino[5], Aurora Ramos[6], X. Du[7], M. Flouris[7], L. Mariani[8], O. Riganelli[8], M. Mobilio[8], A. Shatnawi[8], M. Orru[8], M. Zembra[9]
[1]BT, [2]Ghent University-imec, [3]Nokia Bell-Labs France, [4]DTU Fotonik, [5]Virtual Open Systems, [6]ATOS, [7]OnApp, [8]Unimib, [9]VM2M

Abstract – **Platform-As-A-Service (PaaS) systems offer customers a rich environment in which to build, deploy, and run applications. Today's PaaS offerings are tailored mainly to the needs of web and mobile applications developers, and involve a fairly rigid stack of components and features. The vision of the H2020 5GPPP Phase 2 Next Generation Platform-as-a-Service (NGPaaS) project is to enable "build-to-order" customized PaaSs, tailored to the needs of a wide range of use cases with telco-grade 5G characteristics. This paper sets out the salient and innovative features of NGPaaS and explores the impacts on Operational Support Systems and Business Support Systems (OSS/BSS), moving from fixed centralized stacks to a much more flexible and modular distributed architecture.**

Keywords—5G; PaaS; BSS, OSS, Cloud Native, Microservices

I. INTRODUCTION

A crucial element of emerging software-defined 5G networks is that they must support a very diverse range of services, some having extremely stringent targets of end-to-end latency approaching sub-millisecond, and others involving non-human end user equipment such as autonomous vehicles, IoT sensors, robots, etc. The wide range of diverse features and types of end user equipment foreseen in 5G deployments, will have to be supported at an unprecedented scale, which poses significant challenges for traditional means of deploying telecommunications infrastructure and services.

To make 5G possible, "cloud native" principles from the more scalable and flexible networks that deliver cloud-based services in IT companies need to be adopted. To realize this vision, an alternative model to Infrastructure-as-a-Service (IaaS) must be adopted, derived from the cloud service providers themselves and made by developers for developers, known as *the Platform as a Service (PaaS)* concept [1]. The PaaS model has the potential to deliver network services with higher agility and performance through "ancillary services" - scalability, high availability, state management, controllers, orchestrators, all of which can be provided once by the PaaS.

In essence, an **ideal 5G PaaS** should break the inherent silos that exist between connectivity and computing, by facilitating the *building*, *shipping* and *running* of Virtual Network Functions (VNFs) with "telco-grade" quality, alongside a blend of third-party applications thus creating more versatile and powerful cloud objects. This has significant ramifications not just for the required architecture of NGPaaS, but also the way in which and Operational Support Systems (OSS) and Business Support Systems (BSS) must be re-factored and aligned with the new NGPaaS framework.

This work has been performed in the framework of the NGPaaS project, funded by the European Commission under the Horizon 2020 and 5G-PPP Phase2 programmes, under Grant Agreement No. 761 557.

This paper presents novel insights into this new and exciting technology domain. Section II summarises unique features that underpin the NGPaaS, using direct comparison with existing "State-of-the-Art" PaaSs to elucidate the distinctions; this section also introduces the proposed NGPaaS architecture. Section III focusses on OSS and BSS in terms of how NGPaaS will drive changes to the "legacy" models in existing Telco, Mobile and IoT domains. Centralised and rigid OSS/BSS stacks must be re-factored to become distributed and flexible, enabling modular and recursive support for "built-to-order" PaaSs customised per use case. Section IV concludes the paper.

II. REALIZING THE NGPAAS

A. Existing PaaS State-of-the-Art

The current PaaS ecosystem is extremely rich and diverse, with a number of offerings from public clouds, such as Google Cloud Platform, Amazon Web Services and Microsoft Azure. These PaaSs offer a wide variety of configurations allowing for a large number of technologies to be utilized, such as Containers and Virtual Machines (VMs). Beyond offering Container or VM-based platforms, cloud providers offer other options such as serverless computing, cloud storage, databases and load balancers. Assessing the distinct options offered between the major cloud providers, there is arguably little difference with the services they offer. The choice between cloud providers will therefore boil down to the number of available datacenters, support offerings, hosting costs and other predominantly commercial decisions. A major drawback is that it is simply not viable to create a completely customizable 'build-to-order' PaaS. Put another way, the desired PaaS must match the often numerous, but sometimes limited offerings available from cloud providers.

B. Next Generation PaaS Characteristics

This section will provide an executive summary view of the salient features of NGPaaS which clearly differentiate it from existing "State-of-the-Art" PaaS offerings.

1) ***Microservice-based PaaS Modularity***

The ancillary services offered by the PaaS to build, deploy and run network services must encompass a very broad

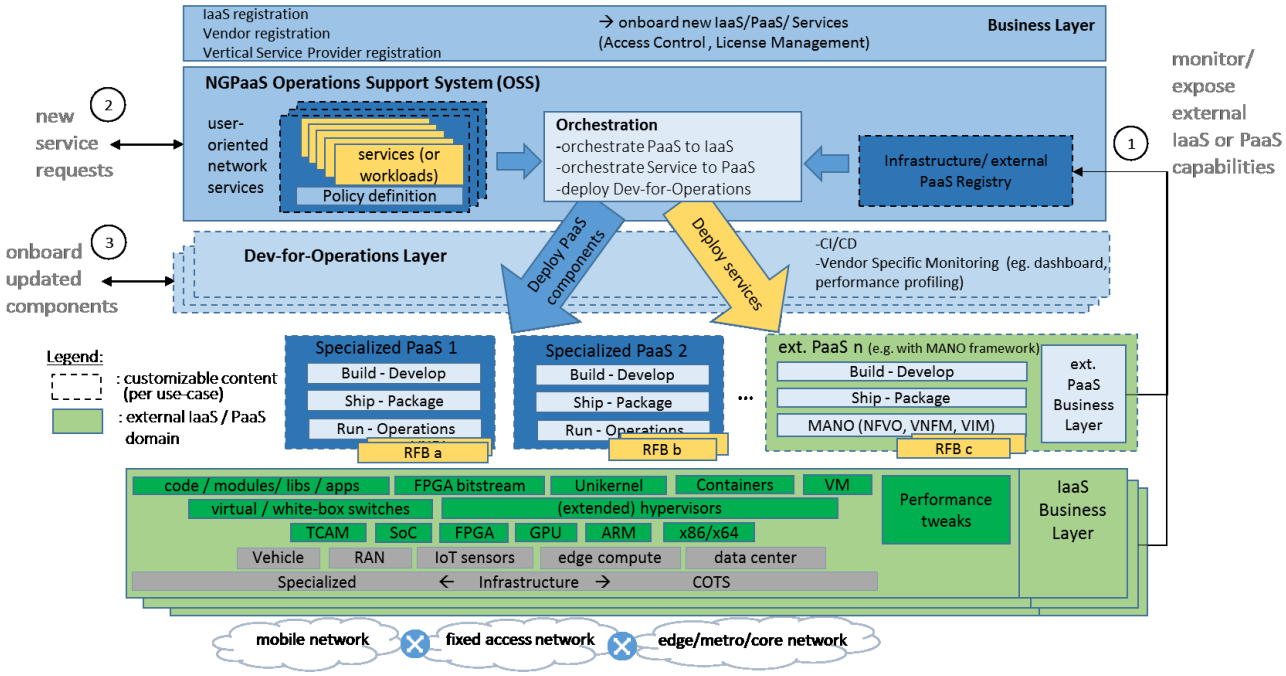

Fig. 1. The OSS in the NGPaaS architecture orchestrates PaaS ancillary services and user-defined workloads to a wide spectrum of virtualized infrastructure. Additionally, the OSS has a supportive role for processing service requests and onboarding new or updated service components.

spectrum of possible virtualization technologies, operational support functions and infrastructure types. To tackle this, we constitute NGPaaS as a *"platform of platforms"*, where specialized PaaSs each target a specific share of the infrastructure. The resultant modular NGPaaS architecture can be subsequently looked at from different viewpoints (Fig. 1):

- **Business View**: The Business Layer on top is used to register all stakeholders who cooperate in the NGPaaS environment. The business-wise affiliation in this layer determines the granted access and execution rights.
- **Design View**: The deployable softwareized components must be delivered in a supporting format, and thoroughly validated in pre-production [2]. The Dev-for-Operations Layer serves this purpose, as will be explained later.
- **Operational View**: The main operational task of the OSS is to deploy requested services and workloads on the appropriate underlying PaaSs. This orchestration will be discussed further in Section III.

Every underlying PaaS and IaaS provider should also offer similar interface capabilities, to fully exploit the advantages of combining many platforms under one NGPaaS, and allowing manageable upgradability of the platform.

2) *Microservice-based Service Modularity*

NGPaaS adopts the microservice based architecture which simplifies complicated software systems by breaking them into sub-components and distributing these components across many computing servers. In a general sense, a *network service* can be implemented likewise, as a chain of Virtualised Network Functions (VNFs) which are in turn deployed as one or more softwarized components mapped to the underlying compute/network/storage resources. These individual microservice based components form the modular building blocks underpinning the network service. As such, they are "Cloud Native VNFs", which process the *workload* imposed by the requested services. We can consider the network service as a *"Service of Services"*, a tree of **decomposable** and **re-usable** softwareized functions, where the leaf nodes need to be mapped to a supporting PaaS for actual deployment. Fig 2 shows a simple illustrative schematic including the "Reusable Function Block" (RFB) - this concept is further expanded in the next section.

3) *Building to Order and Build, Ship, Run*

By implementing NGPaaS as a combination of microservices, we move away from a monolithic locked-in platform with a fixed set of imposed features. Instead, NGPaaS can be customized to choose more freely the functionalities and technologies needed to support a certain business scenario (i.e. use case). Building to order involves the use of components that best match the target service requirements, in terms of business features, performance requirements, service-level agreements, and so on. Following closely the 12-factor cloud-based design principles [3], our modular design model is leveraged by the Reusable Functional Block (RFB), introduced in the 5G-PPP Superfluidity phase-1 project [4]. With NGPaaS, both the PaaS ancillary services and the user-oriented workloads can be decomposed into RFBs and used as building blocks to create a tailored implementation for any use-case. As shown in Fig. 2, both PaaSs and Network Services can be described by RFBs.

An RFB is defined by metadata and an associated execution environment which describes the link between functionality and the infrastructure. Within NGPaaS, the Superfluidity notion of

RFB is extended to allow additional procedures of Build, Ship and Run. Build involves the creation of components, Ship securely transfers the newly-created component to the required execution environment while Run deploys and runs the component on its final target.

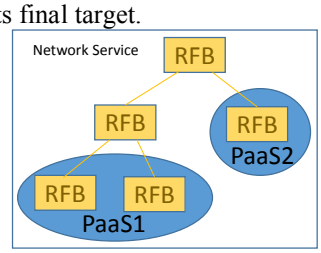

Fig 2: Network Service (or PaaS) decomposed as tree of RFBs.

4) ***Telco-Grade PaaS Features***

A number of significant enhancements to existing PaaS frameworks will be required to support "Telco-grade" features:

- *Specialized Hardware*: In defining the requirements of a "build-to-order" PaaS, it must be stipulated whether the underlying IaaS can be based on general purpose compute, or alternatively specialized hardware is needed to meet much more demanding workloads and associated Service-Level Guarantees (SLGs) such as maximum latency. An example of specialized hardware for such a purpose could be the use of Field Programmable Gate Array (FPGA) offload[5].
- *Custom Hypervisor Features*: Acceleration of packet forwarding can be achieved using specialized, lightweight hypervisors such as the MicroVisor platform[6]. This customized hypervisor technology efficiently optimizes network and storage I/O.
- *Integrated SDN Controller:* In NGPaaS, the SDN control layer allows for smooth integration of heterogeneous network infrastructure and the specific business and orchestration layer for the PaaS. In terms of differentiated "Telco-grade" aspects, this would include aspects such as access/programmability of FPGA-based acceleration units, and application of policy enforcement rules linked to the actual use case.
- *Monitoring, Profiling & Healing:* NGPaaS proposes to extend in situ PaaS monitoring capabilities for the purpose of performance profiling. The capture and analysis of service-specific data metrics can be used to build a profile, e.g. of a specific VNF, and performance profiles can then be tracked to detect changes in behavior and anomalies. To meet Service Level Agreements (SLAs), NGPaaS proposes a two-step self-healing process. The first step predicts failures and localizes the likely responsible resources by exploiting monitored KPIs. The second step activates countermeasures to prevent or heal failures. Self-healing uses online analytics techniques to analyze the KPIs and identify anomalies, while Machine Learning (ML) techniques filter actual anomalies from noise.

These "Telco-grade" PaaS features should be considered optional as required by specific use cases. In line with the "build-to-order" ethos of NGPaaS, the appropriate features are defined as part of customized properties of the PaaS itself, and the service-oriented workloads that run on top.

5) ***Operational Maintenance and Dev-for-Operations***

The preceding section discussed how to overlay "Telco-grade" features into the PaaS domain. At the same time, there are certain operational cloud-oriented practices that can be incorporated into the NGPaaS architecture. The adoption of microservices opens the door to maintenance practices such as partial updates and hot-swapping of components. These methods can greatly improve operational stability if executed properly and permit quicker maintenance cycles. Two main areas that can be upgraded are the platform services, such as the hypervisor or FPGA framework, and application services. In NGPaaS, we aim to allow upgrades of each individual platform, as dictated by the business logic layer. Actual upgrades can be carried out by live upgrade, rolling upgrade or cold upgrade.

While DevOps is adopted in the IT industry to permit closer collaboration between development and operational teams inside a single organization, NGPaaS extends this "in-house" flow to a wider range of stakeholders. The goal is to apply DevOps practices to cross-organizational virtual work-teams, a paradigm we call "Dev-for-Operations". Beyond the IT-oriented requirements usually associated with DevOps, we will support Telco-grade requirements to meet the needs of 5G. To support this, the NGPaaS architecture provides a customizable Dev-for-Operations layer, tailored to each vendor allowing custom access and execution rights (Fig 1). These layer instances can be deployed to allow customizable access to the PaaS where services are deployed, with Continuous Integration/Delivery (CI/CD) enabling automatic integration of the components after validation tests have been executed.

III. OSS RE-FACTORING

A. *Addressing Limitations with Existing OSS/BSS Models*

Legacy monolithic platforms have no simple way of interpreting high level business level requirements, then translating them into scalable and distributed resource and service offerings, especially from the OSS/BSS perspective. They therefore tend to solve all problems by requiring more low-level complexity and lack of flexibility. NGPaaS aims to define a new cloud-stack which galvanizes open collaboration between all stakeholders involved in network service provisioning (vendors, service providers, etc), thus moving away from a hierarchical cloud stack with a fixed set of features, to a modular and distributed stack. Transforming from a 'one-size-fits-all' solution, generates some obvious ramifications for OSS/BSS. Taking Fig.1 as a base, the re-factored OSS is now distributed across two levels:

1) From the over-arching NGPaaS OSS, a two-phased orchestration mechanism arises. The first phase deploys the specialized PaaS plus its ancillary services (e.g. "add-ons" like auto-scaling, monitoring, etc) to a set of allocated resources. The second phase deploys service-oriented workloads onto an already-deployed PaaS.

2) The PaaSs themselves offer ancillary services which enhance the OSS further on the PaaS level (e.g. specialized monitoring or autoscaling).

In the NGPaaS context, we focus on the OSS at the top level. There are three main interfaces which the OSS must support, as illustrated in Fig. 1. Using ❶, the IaaS providers can expose any available infrastructure which can be used by the OSS to deploy a PaaS. The OSS should then first deploy one or more specialized PaaSs, linked to the specific business use-case, on the infrastructure. The interface at ❷, allows a service provider to request a service or impose a workload to the NGPaaS Operator from a use-case tailored catalog. The orchestration mechanism in the OSS must now decompose the request into RFBs and map them to an available PaaS. The interface at ❸ is used complementary to operation, to assist an external vendor. The interface allows custom monitoring, access and execution rights to onboard new or updated service or PaaS components. As explained before, the Dev-for-Operations layer can implement a CI/CD function to validate components before they are on-boarded in the production environment. In this context, the OSS needs to be addressed by the Dev-for-Operations layer. Primarily, the OSS must orchestrate the software component under test, to deploy it in a CI/CD test (an isolated PaaS or IaaS slice should be targeted). Furthermore, metrics gathered by the OSS can be requested by the Dev-for-Operations layer to assess functionality or feedback (filtered) data back to the vendor.

The next section explores and introduces in more detail how NGPaaS's distributed and modular OSS can be practically implemented using Reusable Function Blocks (RFBs).

B. Implementing Distributed & Modular OSS with RFBs

RFBs are by nature highly distributed and designed to be deployed in heterogeneous infrastructure [4]. Fig. 3 describes our RFB-based OSS/BSS model. At the top level, an **R**FB **O**SS/**B**SS **M**aster (ROBM) manages the entire infrastructure and handles BSS requests from the business layer. It runs the initial deployment and acts upon information received from PaaS-specific OSS, named the **R**FB **O**SS/**B**SS **A**gent (ROBA), through a global message bus. This hierarchical design enables new capabilities. Autonomy is maintained at different levels so the ROBA may take its own decisions about its PaaS and react accordingly. The RFB Service layer enables service providers to share architectural components (PaaS-level RFB) from different vendors, which can then work together seamlessly, independent of which company supplies them. High level Service RFBs are delimited by Domains, which are *namespaced* and allow for projects to have a reasonable scope, and not forced to span the entire set of ROBM requirements.

The ROBM is a meta-OSS/BSS in charge of managing top-level NGPaaS OSS/BSS tasks such as business layer policy execution and dynamic inventory supervision. For service deployment, the ROBM checks first the availability of resources in the inventory, starts instantiation of the RFB service domain, the head RFB service descriptor, and then the dedicated ROBA through the global pub/sub bus. The ROBA is a microservice attached to the RFB service descriptor. Service domain deployment is then delegated to the ROBA which is in charge of instantiating PaaS Platform level RFBs, initializing the Execution Environment (EE), the local pub/sub bus, and then deploying RFB leafs. The ROBA communicates locally with domain components using the local pub/sub bus. The ROBA then receives a delegation for doing local OSS supervision and operation tasks as well ensuring local BSS executions. Since our system is fully recursive, the previous scenario can be repeated. As depicted in Fig 3, ROBA can instantiate a service RFB sub-domain with the ROBA2 as the local OSS/BSS agent, delegate sub-domain OSS/BSS tasks, then in turn act as Master for the ROBA2.

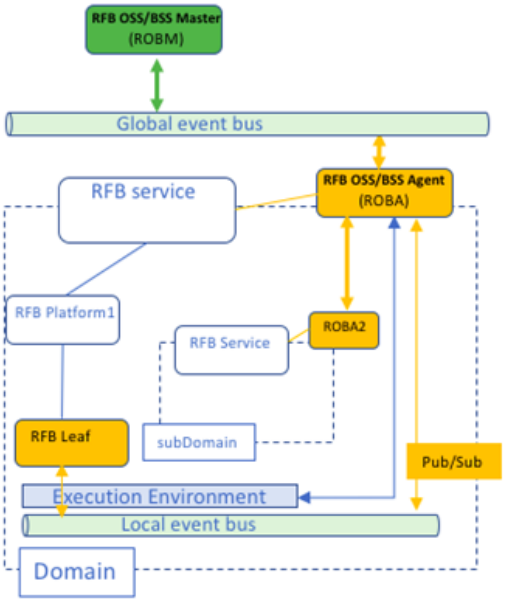

Fig. 3. RFB Hierarchical Architecture Overview

Fig 4 shows the key internal components. The ROBA uses OSS and BSS APIs to communicate with upper and lower OSS/BSS components. A *public* shared interface is used for ROBM-ROBA communication, and ensures inter-ROBA communication and connectivity with external OSS/BSS. At the ROBA Domain level, a *private* API serves to interconnect intra-Domain internal components and to communicate with the EE (the EE proxy registers and connects to the local Execution Environment), while the analytics plugin provides an extension to external analytics microservices. In the ROBM, the "Inventory proxy" registers and connects to the Dynamic Inventory to manage real-time global resource and infrastructure usage. The ROBM then delegates local resource management to ROBAs. The "OSS/BSS proxy" component maintains connectivity with existing legacy OSS/BSS systems.

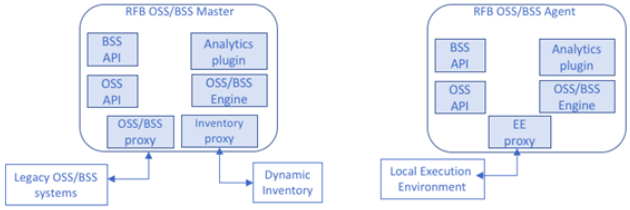

Fig. 4. ROBM (left) and ROBA (right) Internal Architecture

Fig. 5 shows the hierarchical OSS/BSS task distribution. Determining where tasks should be first executed and how to react and where to check in case of failure, can be complex to model with such a distributed and hierarchical system. OSS/BSS tasks are therefore split into three families (Table I). OSS Operation covers deployment and updates, while OSS Supervision covers performance monitoring, health checking and alarm monitoring. BSS Validation covers SLA and KPI validation. As indicated in Table I, task execution should first be initiated locally, then globally, to avoid any scalability and performance issues that could arise by executing globally first.

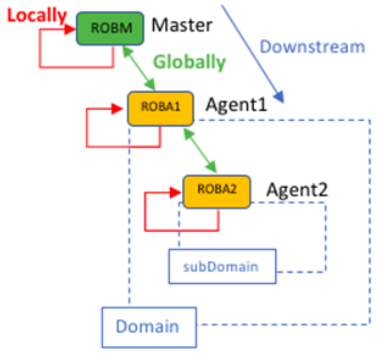

Fig. 5. Hierarchical OSS/BSS Task Distribution

TABLE I. OSS/BSS Task Categorization

	OSS Operation	**OSS Supervision**	**BSS Validation**
Execution	Locally	Locally then Globally	Locally then Globally
Result interpretation	Locally then Globally (downstream)	Locally then Globally	Locally then Globally
Acting upon result	OSS Operation: Locally then Escalate (upstream)	OSS Operation: Locally then Globally	OSS Operation: Locally then Globally

All underlying PaaS agents (ROBAs) should offer similar interface capabilities to fully exploit the advantages of combining many platforms under one common NGPaaS API and allow manageable upgradability of the platform. Finally, from the orchestration perspective, it should be stressed that the framework using our RFB model is natively distributed by design choice. Each orchestration task is first executed locally then delegated to an underlying orchestrator (ROBA). We then provide on each domain an automated system that is aware of its execution environment and responds dynamically and locally to observed changes. This enables fine grained orchestration leveraging domain-based and context-based data extracted in real-time from the local pub/sub bus.

C. Extensive Scope of Recursion Principles in NGPaaS

Recursion is an underlying principle that we leverage at different levels of the NGPaaS architecture. In specific relation to RFBs as detailed in the preceding section, we implement an OODA loop. ***Observe***: we monitor and collect the data. ***Orient*:** we perform some data analysis and build model. ***Decide*:** we plan an action and target some resources. ***Act*:** acting upon the resources, we then observe again. In a broader sense, a recursive structure in 5G can be defined as a design, rule or procedure that can be applied repeatedly [7]. In a network service context, this can either be a specific part of a network service or a repeated part of the deployment platform permitting a service to be built from existing services. A recursive structure in the 5G software architecture can be instantiated and linked repeatedly. It improves scalability, as the same instance can be deployed many times, at different places at the same time.

Recursive orchestration in the NGPaaS OSS, requires extension to interface ❶ in Fig. 1. Not only IaaS, but also third-party PaaS providers can now register with the NGPaaS platform. Similar to IaaS providers, external PaaS providers can expose a set of resources in the form of supported services or workloads to be deployed. A higher-level orchestrator can then offer and combine these exposed third-party services or workloads into its own catalogue. The ability to generically accept *any* external service request, will present notable challenges. Firstly, the requested service software needs to be trusted (or adequately isolated), and secondly the external PaaS must also be trusted, and have appropriate OSS capabilities to deploy and operate any delegated software components. Since an external PaaS can be another NGPaaS platform on its own, a MANO platform (like SONATA, OSM or ONAP) provided by a third-party, or another commercial cloud platform, it will invariably have limited or specialized capabilities regarding available infrastructure and OSS functionality. For practical reasons therefore, it is proposed that only a *pre-defined* set of supporting services would be exposed by the external PaaS.

IV. Conclusion and Further Work

The vision of the H2020 5GPPP Phase 2 Next Generation Platform-as-a-Service (NGPaaS) project is to enable "build-to-order" customized PaaSs, tailored to the needs of a wide range of use cases with Telco-grade 5G characteristics. This paper has explained the salient features of NGPaaS and described the impacts on Operational Support Systems and Business Support Systems (OSS/BSS). A novel architectural framework is under development which moves from fixed centralized stacks to a much more flexible and distributed model. During 2018, these innovations will be further refined and a number of Proof-of-Concept testbeds built to practically demonstrate a range of use cases covering Telco, Mobile and IoT industry verticals.

References

[1] S. Kolb, C. Röck, "Nucleus - Unified Deployment and Management for Platform as a Service", Otto-Friedrich-Universität Bamberg, 2016.

[2] S.Van Rossem,W. Tavernier, D. Colle, M. Pickavet, P. Demeester." Introducing Development Features for Virtualized Network Services." IEEE Communications Magazine. Dec 2017.

[3] dam Wiggins "The Twelve-Factor App" [online].

[4] Superfluidity, "Deliverable D3.1: Final system architecture, programming interfaces and security framework spec..', December 2016.

[5] M. Paolino, S. Pinneterre, S.D. Raho. "FPGA virtualization with accelerators overcommitment for Network Function Virtualization", 2017 International Conference on Reconfigurable Computing and FPGAs.

[6] X. Ragiadakou M. Alvanos J. Chesterfield J. Thomson M. Flouris "Microvisor: A scalable hypervisor architecture for microservers" 2015.

[7] 5G PPP Architecture Working Group, white paper revision 2.0, "View on 5G Architecture", Dec 2017.